

Scalable parallel preconditioners for an open source cardiac electrophysiology simulation package

Miguel O. Bernabeu^{a,*}, David Kay^a

^a*Oxford University Computing Laboratory, Oxford, United Kingdom*

Abstract

The numerical solution of the bidomain equations is a fundamental tool in the field of computational cardiac electrophysiology. The multi-scale nature of the processes involved and the increasing complexity of the tissue and ionic models of interest make the use of High Performance Computing techniques mandatory. The solution of the linear system arising from the Finite Element discretisation of the bidomain equations is the main bottleneck in terms of performance and parallel scalability. In this work we present and evaluate parallel implementations of two problem-specific block preconditioners, namely BD and LDU, in the framework of the open source cardiac electrophysiology simulation package Chaste. Both rely upon providing a reliable and practical method for calculating the action of the inverse of certain matrix subblocks. Our results show that: a) total execution time can be reduced by using different approximation methods for different matrix subblocks, b) there exists a good correlation between the coefficient $\Delta t/h^2$ and the cases where BD and LDU outperform generic Algebraic Multigrid (AMG) preconditioning, and c) when applied to simulations with current state-of-the-art cardiac geometries, BD and LDU are over 3 times faster than AMG preconditioning and show similar scaling.

Keywords: bidomain simulation, parallel preconditioning, open source, Chaste

1. Introduction

The numerical solution of the bidomain equations [1] is a fundamental tool in the field of computational cardiac electrophysiology. The bidomain equations model electrical propagation in cardiac tissue as a multi-scale reaction-diffusion process. Their numerical approximation involves the solution of a system of two partial differential equations (PDEs) describing the evolution of the homogenised intracellular and extracellular potentials, coupled at each point in space to a system of ordinary differential equations (ODEs) describing ion kinetics across the cellular membrane (i.e. the ionic model).

The multi-scale nature of the processes involved and the increasing complexity of the tissue and ionic models of interest make the use of High Performance Computing (HPC) techniques mandatory. When choosing a semi-implicit temporal discretisation of the system [2], one can successfully decouple the solution of the ODEs from the system of PDEs. The solution of the ionic models then becomes embarrassingly parallel and does not compromise

*Corresponding author address: Oxford University Computing Laboratory, Wolfson Building, Parks Rd, Oxford OX1 3QD, United Kingdom
Email addresses: miguel.bernabeu@comlab.ox.ac.uk (Miguel O. Bernabeu), david.kay@comlab.ox.ac.uk (David Kay)

scalability. However, solution of tissue level diffusion (system of PDEs) via the Finite Element Method (FEM) is a tightly coupled problem and often the main parallel bottleneck [3]. Linear systems arising from bidomain FEM discretisation are naturally ill-conditioned and in recent years there has been an increasing interest in the development of efficient preconditioners in order to speed up simulations, both in sequential [4, 5] and in parallel [6, 7].

In previous work [4], we presented two efficient sequential preconditioners for the solution of the bidomain equations on unstructured grids, namely BD and LDU. Both achieved good performance and in most cases outperformed generic preconditioning techniques such as Algebraic Multigrid (AMG). We also proved theoretically and empirically the mesh-independent convergence properties of LDU.

In the current work, we will present parallel implementations of BD and LDU and extend the convergence analysis to other problem characteristics. BD and LDU rely upon providing a reliable and practical method for calculating the action of the inverse of certain matrix subblocks. In [4] we used 1 AMG V-cycle, but in this work we will investigate other options. Unlike previous publications [4, 5, 6], we will not consider AMG a black box, instead we will explore some of the configuration options of one of the most mature libraries for parallel AMG preconditioning: the HYPRE library developed at the Lawrence Livermore National Laboratory. Previous studies of bidomain preconditioning scalability [6, 7] did not go beyond a few hundred processors. We will present results for core counts one order of magnitude larger.

The purpose of this paper is two-fold. First, present two parallel preconditioners for the numerical solution of the bidomain equations. Second, study their efficiency as a function of number of processors, spatial, and temporal discretisation in simplified geometries as well as in realistic 3-D anatomically based geometries.

Section 2 describes the numerical methods employed for the solution of the bidomain equations. The linear solver and preconditioners considered in this work are introduced in Section 3, followed by a description of their parallelisation in Section 4. Section 5 presents the benchmarks used to evaluate performance and scalability. Finally, Section 6 presents the benchmarks results and Section 7 analyses them and summarises the main findings.

2. Numerical solution of the bidomain equations including a passive surrounding medium

2.1. Formulation

We consider bidomain simulations of cardiac tissue contained in a conductive surrounding medium, i.e. the bath. The magnitudes of interest are intracellular and extracellular potential (ϕ_i and ϕ_e), as well as their difference ($V = \phi_i - \phi_e$). We suppose there are two disjoint domains: the tissue Ω and the bath Ω_b , with interface $\partial\Omega$. In this problem ϕ_i , and therefore V , is only defined in Ω , but ϕ_e is defined throughout $\Omega \cup \Omega_b$.

The problem to be solved is: find $V \in H^1(\Omega)$ and $\phi_e \in H^1(\Omega \cup \Omega_b)$ satisfying

$$\chi \left(C \frac{\partial V}{\partial t} + I_{\text{ion}} \right) - \nabla \cdot (\sigma_i \nabla \phi_i) = -I_i^{(\text{vol})}, \quad \text{in } \Omega \quad (1)$$

$$\nabla \cdot (\sigma_i \nabla \phi_i + \sigma_e \nabla \phi_e) = 0, \quad \text{in } \Omega \quad (2)$$

$$\nabla \cdot (\sigma_b \nabla \phi_e) = 0, \quad \text{in } \Omega_b \quad (3)$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u}, V), \quad \text{in } \Omega$$

where σ_b is the bath conductivity, σ_i is the intracellular conductivity tensor, σ_e the extracellular conductivity tensor, χ is the surface-area-to-volume ratio and C is the membrane capacitance per unit area. The vector \mathbf{u} contains cell-level variables, such as ionic concentrations and membrane gating variables, and $I_{\text{ion}} \equiv I_{\text{ion}}(\mathbf{u}, V)$ is the ionic current per unit surface area. Functional forms for I_{ion} and \mathbf{f} are determined by an electrophysiological cell model, examples of which can be found in the CellML repository¹. The source term $I_i^{(\text{vol})}$ is the intracellular stimulus per unit volume. Note, we have used ϕ_i here instead of $V + \phi_e$ in order to simplify the equations.

Appropriate boundary conditions are

$$\mathbf{n} \cdot (\sigma_i \nabla \phi_i) = 0, \quad \text{on } \partial\Omega \text{ (i.e. bath-tissue boundary)} \quad (4)$$

$$\mathbf{n} \cdot (\sigma_b \nabla \phi_e) = I^{(\text{E})}, \quad \text{on } \partial\Omega_b \setminus \partial\Omega \text{ (i.e. external bath boundary)} \quad (5)$$

¹<http://models.cellml.org/>

where \mathbf{n} is the outward-facing unit normal. Here $I^{(E)}$ is a stimulus current per unit area representing an external current flowing into the domain and causing an electrical shock on the tissue surface. The problem is stated without any Dirichlet boundary conditions and therefore ϕ_e and ϕ_i are defined up to a constant. For a solution to exist, the compatibility condition $\int_{\partial\Omega_b \setminus \partial\Omega} I^{(E)} dS = 0$ (i.e. input current is equal to output current) must be satisfied.

2.2. Finite element solution

We now briefly describe the linear system arising from the FEM spatial and semi-implicit temporal discretisation of (1)-(5) (see [2] for a more detailed discussion). Let V^m denote the voltage at time t_m , and similarly with other variables. Let us suppose the first N nodes are contained in the closure of Ω , and the next M nodes are the remaining nodes in the bath ($\bar{\Omega}_b \setminus \bar{\Omega}$). We can write $V^m(x) = \sum_{j=1}^N V_j^m \psi_j(x)$, which has to be understood to apply only for $\mathbf{x} \in \bar{\Omega}$, and $\phi_e^m(x) = \sum_{j=1}^{N+M} \Phi_j^m \psi_j(x)$. Let $\mathbf{V}^m = (V_1^m, \dots, V_N^m)$, $\Phi^m = (\Phi_1^m, \dots, \Phi_{N+M}^m)$ and define $\Phi_{(1)}^m = (\Phi_1^m, \dots, \Phi_N^m)$ and $\Phi_{(2)}^m = (\Phi_{N+1}^m, \dots, \Phi_{N+M}^m)$. The finite element problem is to find \mathbf{V}^{m+1} , $\Phi_{(1)}^{m+1}$, and $\Phi_{(2)}^{m+1}$ such that

$$\begin{bmatrix} \frac{\chi C}{\Delta t} M + K[\sigma_i] & K[\sigma_i] & 0 \\ K[\sigma_i] & \mathcal{K}_{(1,1)} & \mathcal{K}_{(1,2)} \\ 0 & \mathcal{K}_{(2,1)} & \mathcal{K}_{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{m+1} \\ \Phi_{(1)}^{m+1} \\ \Phi_{(2)}^{m+1} \end{bmatrix} = \begin{bmatrix} \frac{\chi C}{\Delta t} M \mathbf{V}^m + \mathbf{c}^m \\ \mathbf{0} \\ \mathbf{d} \end{bmatrix} \quad \begin{matrix} \text{size } N \\ \text{size } N \\ \text{size } M \end{matrix}$$

where

$$\begin{aligned} M_{jk} &= \int_{\Omega} \psi_j \psi_k d^3 \mathbf{x} \\ (K[\sigma])_{jk} &= \int_{\Omega} \nabla \psi_j \cdot (\sigma \nabla \psi_k) d^3 \mathbf{x} \\ c_j^m &= - \int_{\Omega} (I_i^{(\text{vol})}(t_{m+1}) + \chi I_{\text{ion}}(\mathbf{u}^m, V^m)) \psi_j d^3 \mathbf{x}. \\ d_j &= \int_{\partial\Omega_b \setminus \partial\Omega} I^{(E)} \psi_{j+N} dS. \end{aligned}$$

and $\mathcal{K}_{(a,b)}$ are the subblocks of a large $N + M$ by $N + M$ stiffness matrix \mathcal{K}

$$\mathcal{K}_{jk} = \int_{\Omega} ((\sigma_i + \sigma_e) \nabla \psi_k) \cdot \nabla \psi_j d^3 \mathbf{x} + \int_{\Omega_b} (\sigma_b \nabla \psi_k) \cdot \nabla \psi_j d^3 \mathbf{x},$$

In practice, for implementation reasons, we introduce a set of dummy voltage values at the bath nodes, V_{N+1}, \dots, V_{N+M} , and add the equations

$$V_j = 0, \quad j = N + 1, \dots, N + M,$$

so that the linear system becomes of size $(2N + 2M)$.

Letting $\mathbf{V}_{(1)}^m = \mathbf{V}^m$, and $\mathbf{V}_{(2)} = (V_{N+1}, \dots, V_{N+M})$ (i.e. the vector of dummy values), and letting I_M denote the M by M identity matrix, we have

$$\begin{bmatrix} \frac{\chi C}{\Delta t} M + K[\sigma_i] & 0 & K[\sigma_i] & 0 \\ 0 & I_M & 0 & 0 \\ K[\sigma_i] & 0 & \mathcal{K}_{(1,1)} & \mathcal{K}_{(1,2)} \\ 0 & 0 & \mathcal{K}_{(2,1)} & \mathcal{K}_{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{(1)}^{m+1} \\ \mathbf{V}_{(2)} \\ \Phi_{(1)}^{m+1} \\ \Phi_{(2)}^{m+1} \end{bmatrix} = \begin{bmatrix} \frac{\chi C}{\Delta t} M \mathbf{V}_{(1)}^m + \mathbf{c}^m \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{d} \end{bmatrix} \quad \begin{matrix} \text{size } N \\ \text{size } M \\ \text{size } N \\ \text{size } M \end{matrix} \quad (6)$$

3. The preconditioned conjugate gradient algorithm

For notational ease, we will rewrite (6) as

$$\mathcal{A} \mathbf{x} = \begin{bmatrix} A_1 & B^T \\ B & A_2 \end{bmatrix} \mathbf{x} = \mathbf{b} \quad (7)$$

with blocks $A_1 = \begin{bmatrix} \frac{\chi^C}{\Delta t} M + K[\sigma_i] & 0 \\ 0 & I_M \end{bmatrix}$, $B = \begin{bmatrix} K[\sigma_i] & 0 \\ 0 & 0 \end{bmatrix}$ and $A_2 = \mathcal{K}$.

\mathcal{A} is symmetric and positive semi-definite. In realistic 3D simulations, \mathcal{A} is also very large and sparse. Therefore Conjugate Gradient (CG) is often the linear solver of choice [2, 5, 6]. The reader can refer to [2] for a discussion on how the non-uniqueness of the solution is handled. Good CG performance is known to be subject to the effectiveness of the preconditioner used. We will consider both generic full-matrix AMG preconditioning (see [8] for details) and problem-specific block preconditioning techniques for the solution of (7).

We define the preconditioner

$$\mathcal{P}_{LDU}^{-1} = \begin{bmatrix} I & -A_1^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BA_1^{-1} & I \end{bmatrix} \quad (8)$$

where A_2^{-1} represents the pseudo-inverse of the singular matrix A_2 . This inverse is not computed explicitly but approximated and acts upon a consistent right-hand side. Along with this choice, the computationally cheaper approximation

$$\mathcal{P}_{BD}^{-1} = \begin{bmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{bmatrix} \quad (9)$$

is also considered. The effectiveness of the proposed preconditioners rely upon providing an effective and practical method for calculating the action of the inverse of certain matrix subblocks.

4. Solver parallelisation and tuning

4.1. Linear system parallelisation

Parallelising the assembly and solution of (6) should not be seen as two independent problems. The effective parallelisation of the whole bidomain solver requires considering how design decisions taken for some portion of the solver will affect other stages. As part of the assembly of \mathbf{c} in (6) we need to solve $I_{\text{ion}}(\mathbf{u}^n, V^n)$ at each grid point. In order to achieve good load-balance, we evenly distribute the number of grid points among the available processors. With the original data layout proposed in (6) (and assuming a row-wise distribution of the matrix), processors owning rows $N + M + 1$ to $2N + 2M$ would have to communicate values I_{ion} computed locally to the processors owning the first N rows for them to assemble \mathbf{c} . Therefore, it makes sense to rearrange the equations in way that \mathbf{c} is assembled from values of I_{ion} computed locally avoiding communication. Similarly we would like to have all the processors cooperating in the evaluation of $\frac{\chi^C}{\Delta t} M \mathbf{V}^m$ and not only those owning the first N rows. In order to satisfy the previous two requirements, we decided to interleave the rows of \mathcal{A} and \mathbf{b} in a way that all the processors contribute equally and with minimum degree of communication to the assembly of \mathbf{b} . The same permutation is applied to the columns of \mathcal{A} and the rows of \mathbf{x} so that the symmetry is preserved. This process can be summarised as applying the following permutation to (7):

$$Q\mathcal{A}Q^T Q\mathbf{x} = Q\mathbf{b} \quad (10)$$

with the orthogonal permutation matrix $Q = [q_{ij}] \in \mathbb{R}^{(N+M) \times (N+M)}$

$$q_{ij} := \begin{cases} 1, & i \text{ is even} \wedge j = \frac{i+n}{2} \\ 1, & i \text{ is odd} \wedge j = \frac{i+1}{2} \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

This is how (10) would look like for a bidomain simulation in a domain made of two grid points and with 2 processors (solid line indicates border between processors):

$$\begin{pmatrix} V_{11} & \phi_{11} & V_{12} & \phi_{12} \\ V_{31} & \phi_{31} & V_{32} & \phi_{32} \\ V_{21} & \phi_{21} & V_{22} & \phi_{22} \\ V_{41} & \phi_{41} & V_{42} & \phi_{42} \end{pmatrix} \begin{pmatrix} V_1 \\ \phi_1 \\ V_2 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ b_2 \\ b_4 \end{pmatrix} \quad (12)$$

where $\{V, \phi\}_{ij}$ is the coefficient associated with unknown $\{V, \phi\}_j$ in the i -th equation.

4.2. Preconditioners parallelisation

In every CG iteration k the following preconditioning operation is performed

$$z^k = M^{-1} r^k \quad (13)$$

where r^k is the residual of the solution, M^{-1} is the preconditioner and z^k is used in the computation of the next search direction. Note that depending on the choice of preconditioner this may or may not involve an explicit matrix-vector product but the application of a linear operator. In this section, we will consider $M^{-1} = \mathcal{P}_{BD}^{-1}$ but the discussion can be easily extended to \mathcal{P}_{LDU}^{-1} . For performance reasons, and assuming that $r^k = (r_V^k, r_{\phi_e}^k)^T, r_{\{V, \phi_e\}}^k \in \mathbb{R}^{N+M}$ and similarly with z^k , (13) can be rewritten as

$$z_V^k = A_1^{-1} r_V^k \quad (14)$$

$$z_{\phi_e}^k = A_2^{-1} r_{\phi_e}^k \quad (15)$$

In order to ensure good parallel scalability in the application of \mathcal{P}_{BD}^{-1} one has to minimise the amount of data to be exchanged between processors for the assembly of data structures r_k^V and $r_k^{\phi_e}$ from r_k (scatter operation) and z_k from z_k^V and $z_k^{\phi_e}$ (gather operation). We will now show how our underlying data structures ensure this. If we have a closer look at how (13), (14), and (15) are distributed (based on the data layout of model problem (12))

$$\begin{pmatrix} z_{V_1}^k \\ z_{\phi_1}^k \\ z_{V_2}^k \\ z_{\phi_2}^k \end{pmatrix} = M^{-1} \begin{pmatrix} r_{V_1}^k \\ r_{\phi_1}^k \\ r_{V_2}^k \\ r_{\phi_2}^k \end{pmatrix}, \quad \begin{pmatrix} z_{V_1}^k \\ z_{V_2}^k \end{pmatrix} = A_1^{-1} \begin{pmatrix} r_{V_1}^k \\ r_{V_2}^k \end{pmatrix}, \quad \begin{pmatrix} z_{\phi_1}^k \\ z_{\phi_2}^k \end{pmatrix} = A_2^{-1} \begin{pmatrix} r_{\phi_1}^k \\ r_{\phi_2}^k \end{pmatrix}$$

it can be seen that intermediate data structures can be assembled with data local to each processor, therefore avoiding communication.

If we were not using the interleaved data layout described in Section 4.1, r_V would be entirely owned by processors owning the first $M + N$ equations and r_{ϕ} by the rest. One could think that (14) and (15) can be executed independently and therefore data does not need to be interchanged with the other half of the processors. Later we will see how different inverse approximations are used for (14) and (15) yielding different computational cost. Therefore, decoupling their computation would compromise load balancing. For \mathcal{P}_{LDU}^{-1} the situation is even worse because (14) needs to be solved twice per CG iteration.

5. Benchmarks

In this section we describe the benchmarks designed to evaluate performance and parallel scalability of the preconditioners presented in Section 3.

5.1. Preconditioners tuning

In [4], we used one AMG V-cycle for the approximation of A_1^{-1} and A_2^{-1} in BD and LDU. We would like to study alternatives for the approximation of A_1^{-1} and A_2^{-1} . This is motivated by the fact that factorisation-based preconditioners are faster per required iteration. We also want to investigate domain decomposition techniques such as Block Jacobi (BJ) or Additive Schwarz Method (ASM) for their obvious benefits in terms of communication reduction. Multiple combinations of these techniques are compared for the solution of a benchmark consisting of 2 ms of bidomain simulation on a simplified version of the anatomically-based ventricular model by Bishop et al. [9]. Tissue is stimulated with a 0.5 ms long extracellular shock of magnitude $-11 \times 10^3 \mu\text{A}/\text{cm}^2$ delivered at $t = 0$ ms through electrode plates located at the boundaries of the bath parallel to the sagittal plane. The cell model used is Luo Rudy 1991 [10] integrated with a backward Euler numerical scheme. Both PDE and ODE timesteps are 0.01 ms.

5.2. Δt - and h -dependence analysis

We will extend the mesh-dependence convergence analysis presented in [4] by evaluating how Δt affects convergence of the preconditioners presented in Section 3. For this purpose, we generated a set of five 3D tissue slabs immersed in conductive bath. The original volume is a $11 \times 11 \times 11$ mm cube meshed as a regular grid with constant distance between nodes along each dimension $h \in \{1, 0.5, 0.25, 0.125, 0.0625\}$ mm. Nodes in the internal $7 \times 7 \times 7$ mm volume are defined as tissue and the rest as bath. Tissue is stimulated with a 1 ms long extracellular shock of magnitude $-11 \times 10^3 \mu\text{A}/\text{cm}^2$ delivered at $t = 0$ ms through electrode plates located at the boundaries of the bath in the yz plane. Stimulation is applied for long enough to trigger depolarisation in the area of tissue closer to the negative electrode. After that, electrodes are switched off and a planar wave travels along the x direction. The cell model used is Luo Rudy 1991 [10], integrated with a backward Euler numerical scheme. PDE timesteps considered are 0.1, 0.05, and 0.01 ms. ODE timestep is 0.01 ms.

In order to also study parallel scaling of the preconditioners, the following combinations of number of processors and discretisations were considered: i) $p = 16$, $h \in \{1, 0.5, 0.25\}$ mm, ii) $p = 128$, $h \in \{0.5, 0.25, 0.125\}$ mm, and iii) $p = 1024$, $h \in \{0.25, 0.125, 0.0625\}$ mm. This choice ensures that the number of nodes per processor stays almost constant for the three experiments run no matter the value of p .

5.3. Realistic 3D simulations strong scaling

In order to evaluate the techniques presented in realistic 3D cardiac models, we designed a benchmark with the anatomically-based ventricular geometry by Bishop et al. [9]. This is the rabbit tetrahedral mesh with highest level of detail currently available, with an average edge length of $125\mu\text{m}$. It consists of approximately 6.45M grid points and 38.3M tetrahedral elements. Bidomain activity is simulated for 5 ms. In order to elicit propagation, a 0.5 ms long square extracellular shock of magnitude $-11 \times 10^3 \mu\text{A}/\text{cm}^2$ is delivered at $t = 0.2$ ms through electrode plates located at the boundaries of the bath parallel to the saggittal plane. The cell model used is Luo Rudy 1991 [10], integrated with a backward Euler numerical scheme. PDE timestep is 0.1 ms and ODE timestep is 0.01 ms.

6. Results

All the simulations presented in this work were run with the open source cardiac simulation environment Chaste [11]. The following parameters were used in equations (1)–(5): $\chi = 1400 \text{ cm}^{-1}$, $C = 1.0 \mu\text{F}/\text{cm}^2$, $\sigma_i = 8.0 \text{ mS}/\text{cm}$, $\sigma_i = \text{diag}(1.7, 0.19, 0.19) \text{ mS}/\text{cm}$, and $\sigma_e = \text{diag}(6.2, 2.4, 2.4) \text{ mS}/\text{cm}$, where $\text{diag}(x, y, z)$ is a 3×3 diagonal matrix with values x, y, z along the diagonal. Chaste uses the implementation of CG, sequential ILU(0) factorisation, and BJ and ASM preconditioning found in PETSc 3.0.0-p8 with no changes to the default configuration. Parallel AMG preconditioning and parallel ILU(0) factorisation is performed with the BoomerAMG and EUCLID algorithms found in the HYPRE library version 2.4.0b (through its PETSc interface) We choose this AMG implementation for its maturity in terms of parallel performance [12] and for having being successfully applied to the solution of the bidomain equations before [4, 6]. BoomerAMG is a sophisticated algorithm with a large number of configuration options that can have a real impact in performance. A complete exploration of the parameter configuration space is beyond the scope of this work. However, we tuned the parameters considered to have the greatest impact in parallel performance: strong connectivity threshold, coarsening algorithm, smoother and interpolation algorithm.

The impact of strong connectivity threshold in BoomerAMG performance for the solution of the bidomain equations has been studied before. Therefore, we use the value of 0.0 reported as optimal in [6]. From the available smoothers, we chose Hybrid Gauss-Seidel based on the scalability results presented in [12]. In that work, the authors prove that Hybrid Gauss-Seidel scales well provided that the relative weight of the off-diagonal block of a row-based partitioned system matrix, θ , is strictly larger than 1 [12, Definition (5.3)] (not to be confused with the strong connectivity threshold also referred as θ in [6]). Several coarsening algorithms are compared in Section 6.1. Interpolation algorithm is set to BoomerAMG's default: `classical`, since experiments with alternative configurations did not yield any benefit².

Left preconditioning is used throughout this work. In order to make our comparisons as fair as possible, we configure PETSc to use unpreconditioned residual for convergence testing. This is known to introduce an overhead

²Results not shown here, available upon request

when evaluating the convergence condition. A relative tolerance of 10^{-10} and PETSc's default stop criteria is used. Preconditioners and benchmarks will be available in Chaste release 2.2 (www.comlab.ox.ac.uk/chaste).

All the simulations presented were run in HECToR Phase 2a. This is a Cray XT4 system with 3072 compute nodes (at the time of writing). Each compute node is a AMD 2.3 GHz Opteron Barcelona quad-core. Each quad-core socket shares 8 GB of memory and a Cray SeaStar2 chip router with 6 links which are used to implement a 3D-torus network topology.

6.1. Preconditioners tuning

Table 1 compares different choices of block inverse approximation for the benchmark described in Section 5.1 run with 8 processors.

Table 1: Total linear system solution time and average number of iterations for benchmark described in Section 5.1. BJ(xxx) means BJ domain decomposition with xxx preconditioning at each block and similarly with ASM(yyy). AMG(yyy) means 1 AMG V-cycle with yyy coarsening.

A_1^{-1} approximation	A_2^{-1} approximation	total CG time (s)	average CG iterations
AMG(Falgout)	AMG(Falgout)	651	15.23
BJ(ILU(0))	BJ(ILU(0))	–	aborted after 200
BJ(ILU(0))	BJ(AMG(Falgout))	2340	63.48
BJ(ILU(0))	ASM(AMG(Falgout))	4230	30.13
BJ(ILU(0))	AMG(Falgout)	518	11.58
BJ(ILU(0))	AMG(Ruge-Stueben)	498	11.25
BJ(ILU(0))	AMG(PMIS)	464	21.55
BJ(ILU(0))	AMG(HMIS or CLIP)	431	19.37
ILU(0)	AMG (HMIS)	406	17.46

6.2. Δt - h -dependence

Table 2, 3, and 4 report average number of iterations per solve and total linear system solve time (over the whole simulation) for the benchmark described in Section 5.2.

Table 2: Δt - h -dependence benchmark with $p = 16$. (a) average number of iterations, (b) linear solve execution time (s)

Δt	h			precond
	1mm	0.5mm	0.25mm	
0.1ms	12.07	13.3	14.72	AMG
	13.18	16.15	19.51	BD
	12.14	13.97	16.17	LDU
0.05ms	11.42	12.37	13.64	AMG
	12.01	14.63	17.57	BD
	11.4	13.07	14.55	LDU
0.01ms	9.93	10.65	11.38	AMG
	10.13	11.79	13.66	BD
	9.85	11.29	12.31	LDU

(a)

Δt	h			precond
	1mm	0.5mm	0.25mm	
0.1ms	4.19	8.24	38.26	AMG
	4.61	8.57	30.37	BD
	4.76	8.38	28.44	LDU
0.05ms	7.6	15.34	71.03	AMG
	8.4	15.54	54.88	BD
	8.96	15.66	51.21	LDU
0.01ms	33.1	66.19	297.99	AMG
	35.48	62.71	215.89	BD
	38.7	67.62	219.44	LDU

(b)

6.3. Strong scaling

Table 5 reports average number of iterations per solve and total linear system solve time (over 50 solves) for the benchmark described in Section 5.3. Figure 1 plots parallel scalability.

Table 3: Δt - h -dependence benchmark with $p = 128$. (a) average number of iterations, (b) linear solve execution time (s)

Δt	h			precond
	0.5mm	0.25mm	0.125mm	
0.1ms	14.93	16.3	34.84	AMG
	17.63	21.67	23.92	BD
	15.23	18.06	20.76	LDU
0.05ms	13.92	15.65	27.88	AMG
	15.83	19.4	21.63	BD
	14.26	16.34	18.23	LDU
0.01ms	12.36	12.99	13.37	AMG
	12.67	15.14	16.94	BD
	12.28	13.79	14.17	LDU

(a)

Δt	h			precond
	0.5mm	0.25mm	0.125mm	
0.1ms	13.02	17.89	161.65	AMG
	14.94	22.62	52.85	BD
	14.34	20.94	52.77	LDU
0.05ms	22.07	34.48	252.63	AMG
	26.8	40.78	95.94	BD
	26.79	38.02	92.63	LDU
0.01ms	98.78	143.91	471.04	AMG
	108.03	164.7	445.46	BD
	115.65	171.84	426.52	LDU

(b)

Table 4: Δt - h -dependence benchmark with $p = 1024$. (a) average number of iterations, (b) linear solve execution time (s)

Δt	h			precond
	0.25mm	0.125mm	0.0625mm	
0.1ms	18.73	35.43	61.18	AMG
	23.45	26	30.79	BD
	19.78	22.78	27.98	LDU
0.05ms	17.43	28.42	57.97	AMG
	21.19	23.44	27.63	BD
	18	20.17	24.62	LDU
0.01ms	14.84	15.57	40.77	AMG
	16.6	18.39	21.42	BD
	15.25	15.72	18.33	LDU

(a)

Δt	h			precond
	0.25mm	0.125mm	0.0625mm	
0.1ms	24.03	64.68	304.02	AMG
	31.15	44.12	90.28	BD
	28.75	40.96	92.87	LDU
0.05ms	44.7	102.9	576.27	AMG
	56	77.24	159.71	BD
	52.67	72.47	161.99	LDU
0.01ms	199.24	279.1	2114.35	AMG
	245.55	336.44	670.58	BD
	247.05	316.19	652.76	LDU

(b)

Table 5: Realistic 3D simulation benchmark. (a) Average CG iterations, (b) Total CG time (s)

p	AMG	BD	LDU
128	109.14	39.92	36.32
256	109.5	39.86	36.2
512	110.22	40.58	37.06
1024	110.62	42.86	39.2

(a)

p	AMG (s)	BD (s)	LDU (s)
128	1310.36	221.48	240.58
256	762.62	140.04	150.46
512	447.43	102.37	109.29
1024	328.1	93.99	97.68

(b)

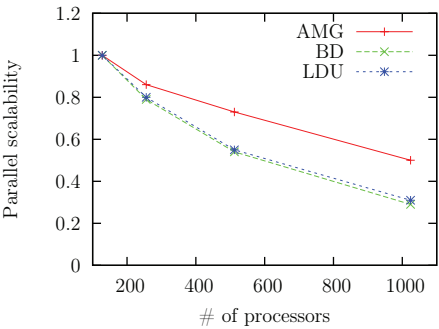


Figure 1: Realistic 3D simulation benchmark strong scaling comparison. Scalability is defined with respect to $p = 128$.

7. Discussion and conclusions

In this work we present and evaluate parallel implementations of two preconditioners proposed in [4]. In sequential, both show better performance than generic preconditioning techniques such as ILU(0) or AMG. However, their parallelisation is challenging because they potentially require data migration for the assembly of intermediate data structures, which can compromise their scalability. Our implementation uses a data partitioning scheme that ensures no data migration when assembling these intermediate data structures. This guarantees that parallel scalability is only restricted by the underlying inverse approximation algorithms and therefore future scalability improvements to them will automatically improve BD and LDU without need of recoding.

In previous work, we considered 1 AMG V -cycle for the approximation of the action of A_1^{-1} and A_2^{-1} with HYPRE's BoomerAMG algorithm default configuration. In this work, we investigated other options. Table 1 shows how using ILU(0) or block jacobi with ILU(0) for the approximation of A_1^{-1} slightly increases the average number of iterations but reduces total execution time. This is not the case with A_2^{-1} , where the iteration count increases dramatically with approximations different from 1 AMG V -cycle. Table 1 also shows how the choice of coarsening algorithm has an important impact on performance. In summary, we reduced by 38% the total linear system solution time of the benchmark presented in Section 5.1 by using ILU(0) for A_1^{-1} approximation and 1 AMG V -cycle with HMIS coarsening for A_2^{-1} compared with the original setup (1 AMG V -cycle with Falgout coarsening for both approximations). The downside to this improvement is that LDU loses the mesh-independent convergence properties proved in [4]. Therefore, we expect that there will be a cutoff value of h below which approximating A_1^{-1} with AMG will reduce overall execution time. This has been investigated and for the finest discretisation considered in this work $h = 0.0625\text{mm}$ (which is one order of magnitude finer than current state-of-the-art cardiac models) approximating A_1^{-1} with ILU(0) is still between 19% and 43% faster (depending on the choice of Δt) than using 1 AMG V -cycle.

We also extended our original h -dependence analysis with a combined h -/ Δt -dependence analysis. Results show a good correlation between large values of $\Delta t/h^2$ and the situations where BD and LDU outperform AMG. This is due to the fact that the condition number of $\mathcal{P}^{-1}\mathcal{A}$ is a function of $\Delta t/h^2$ when mesh-dependent preconditioners are considered [13]. Therefore, the number of iterations required by AMG will grow unbounded as the condition number increases. In contrast, LDU iteration count either: i) converges to an asymptotic bound when 1 AMG V -cycle is used for the approximation of both subblocks [4], or ii) grows at a smaller rate when the efficacy of the first block inverse approximation is relaxed (see Table 4). Table 6 gives the value of the coefficient for all the combinations of Δt and h considered and highlights in dark grey cases where BD and LDU outperform AMG. In our experiments, this happens for values of $\Delta t/h^2$ greater than 2.5. For values below 2.5, the system is not so ill-conditioned and all the preconditioners considered keep iteration count low. In that scenario, the slightly lower computational cost per iteration of AMG makes it faster than BD and LDU.

Table 6: Coefficient $\Delta t/h^2$ for all the combinations of Δt (ms) and h (mm) considered. In dark grey, cases where BD and LDU outperform AMG for any number of processors. In light grey, configurations where this only happens for low core counts

$\Delta t \backslash h$	1	0.5	0.25	0.125	0.0625
0.1	0.1	0.4	1.6	6.4	25.6
0.05	0.05	0.2	0.8	3.2	12.8
0.01	0.01	0.04	0.16	0.64	2.56

Interesting exceptions are the combinations highlighted in light grey in Table 6 and in particular $h = 0.25\text{mm}$, $p = 16$ (Table 2) where AMG has a lower iteration count than LDU, but the latter is around 27% faster than the former. However, this result does not hold for $p = 128$ (Table 3) and $p = 1024$ (Table 4) as one would expect for a value of $\Delta t/h^2$ below the given bound. We believe that in this case the number of degrees of freedom per processor also has an impact on performance. More precisely, in scenarios where a large number of degrees of freedom are stored in each processor, the fact that LDU (and BD) works with two subproblems half the size of the one considered by AMG may have a positive impact in memory utilisation reducing cache misses and/or page fault rates. Effectively shifting the empirical bound given before. Further profiling needs to be carried out to confirm this hypothesis, though.

In agreement with [12], we see a slight increase in the average iteration count with the number of processors for any fixed problem. This is due to a decrease in the relative weight of the off-diagonal block θ . Figure 2 plots average

number of iterations for $h = 0.25$ mm, $\Delta t = 0.1$ ms and 16, 128, and 1024 processors. Similar results are observed in the realistic 3D benchmark.

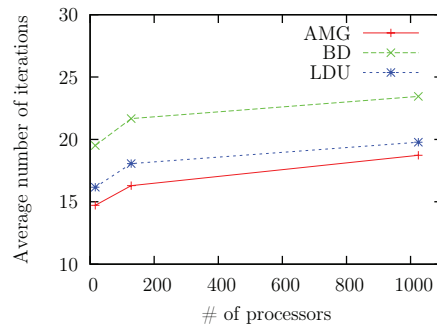


Figure 2: Average number of iterations for benchmark described in Section 5.2, $h = 0.25$ mm, and $\Delta t = 0.1$ ms

Finally, strong scaling of the three methods (Figure 1) is good although not excellent. The reasons are: i) the increase in iteration count with number of processors described above, and ii) the tightly-coupled nature of the approximation algorithms. Furthermore, we see that strong scaling is slightly better for AMG than for BD and LDU. This is due to the fact that in AMG a single problem of size $2M + 2N$ is considered while in BD and LDU two subproblems of $M + N$ are solved. This reduces the number of degrees of freedom per processor for a fixed problem size and therefore scalability tails off faster. Nevertheless, total linear system solution time with BD and LDU is still over 3 times faster than with AMG for the range of core counts considered in the strong scaling study.

Acknowledgements

The work of M.O.B., part of the preDiCT project, was supported by a grant from the European commission DG-INFSo - grant number 224381. The authors would like to thank Dr Martin Bishop for providing some of the geometries used in this study, Dr Nicholas Wilson from Fujitsu Laboratories Europe, HECToR's helpdesk and the Chaste development team (<http://web.comlab.ox.ac.uk/chaste/theteam.html>).

- [1] C. S. Henriquez, Simulating the electrical behavior of cardiac tissue using the bidomain model., *Crit Rev Biomed Eng* 21 (1) (1993) 1–77.
- [2] P. Pathmanathan, M. O. Bernabeu, R. Bordas, J. Cooper, A. Garny, J. M. Pitt-Francis, J. P. Whiteley, D. J. Gavaghan, A numerical guide to the solution of the bidomain equations of cardiac electrophysiology, *Progress in Biophysics and Molecular Biology* 102 (2-3) (2010) 136 – 155.
- [3] J. Southern, N. Wilson, M. O. Bernabeu, J. Pitt-Francis, Chaste: Scalable high-performance simulation of cardiac electrophysiology, in: *1st Virtual Physiological Human Conference - VPH 2010*, 2010.
- [4] M. Bernabeu, P. Pathmanathan, J. Pitt-Francis, D. Kay, Stimulus protocol determines the most computationally-efficient preconditioner for the bidomain equations, *Biomedical Engineering, IEEE Transactions on PP* (99) (2010) 1 –1.
- [5] M. Pennacchio, V. Simoncini, Algebraic multigrid preconditioners for the bidomain reaction–diffusion system, *Appl. Numer. Math.* 59 (12) (2009) 3033–3050.
- [6] G. Plank, M. Liebmman, R. W. dos Santos, E. J. Vigmond, G. Haase, Algebraic multigrid preconditioner for the cardiac bidomain model, *IEEE Trans. Biomed. Eng* 54 (2007) 585–596.
- [7] L. F. Pavarino, S. Scacchi, Multilevel additive schwarz preconditioners for the bidomain reaction-diffusion system, *SIAM J. Sci. Comput.* 31 (1) (2008) 420–443.
- [8] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley & Sons, 1992, reprinted by www.MGNet.org.
- [9] J. Bishop, G. Plank, R. A. Burton, J. E. Schneider, D. J. Gavaghan, V. Grau, P. Kohl, Development of an Anatomically-Detailed MRI-Derived Rabbit Ventricular Model and Assessment of its Impact on Simulation of Electrophysiological Function, *Am J Physiol Heart Circ Physiol* (2009) 00606.2009.
- [10] C.-H. Luo, Y. Rudy, A model of the ventricular cardiac action potential - depolarisation, repolarisation and their interaction, *Circ Res* 68 (1991) 1501–1526.
- [11] J. Pitt-Francis, P. Pathmanathan, M. O. Bernabeu, R. Bordas, J. Cooper, A. G. Fletcher, G. R. Mirams, P. Murray, J. M. Osborne, A. Walter, S. J. Chapman, A. Garny, I. M. van Leeuwen, P. K. Maini, B. Rodríguez, S. L. Waters, J. P. Whiteley, H. M. Byrne, D. J. Gavaghan, Chaste: A test-driven approach to software development for biological modelling, *Computer Physics Communications* 180 (12) (2009) 2452 – 2471, 40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures.
- [12] A. Baker, R. Falgout, T. Kolev, U. Yang, Multigrid smoothers for ultra-parallel computing, *Tech. Rep. LLNL-JRNL-435315*, Lawrence Livermore National Laboratory (2010).
- [13] A. J. Wathen, Realistic Eigenvalue Bounds for the Galerkin Mass Matrix, *IMA Journal of Numerical Analysis* 7 (4) (1987) 449–457.